Major and Minor Scales

Mapping Major and Minor Scales as Flex Sensors are Bent

```
SerialPort.closeAll;

(
/*
SerialPort.closeAll;
SerialPort.devices;
s.quit;
*/

//initial setup
~port=SerialPort("/dev/tty.usbmodem1411", 9600);
~val = [];
~min = 615;
~max = 830;
~a_min = 592;
~b_min = 625;
~c_min = 630;
~a_max = 812;
~b_max = 813;
~c_max = 834;
d = Dictionary.new;
d.add(\a -> 0);
d.add(\b -> 0);
d.add(\c -> 0);

Tdef(\getdata,{
    loop{
        var ascii;
        ascii=~port.next;
        case
        {ascii==nil} {nil}
        {ascii.asAscii.isAlpha}
        {
            if(
                ~val.size>0,
                {
                    d[ascii.asAscii.asSymbol] = ~val.convertDigits;
                    ~val = [];
                }
            );
        }
        {ascii.asAscii.isDecDigit} {~val=~val.add(ascii.asAscii.digit)}
        {true}{nil};

        //d.postln;
        0.0001.wait;
    };

});

s.waitForBoot({
```

```
~whiddittwo= Buffer.read(s,"/Users/Casey/Music/MusicGlove/WhidditTwo.wav");
~whiddittwoM1= Buffer.readChannel(s,"/Users/Casey/Music/MusicGlove/WhidditTwo.wav", channels:[0


//FLANGER
    SynthDef.new(\flanger,{
        arg out=0, buf=0, density=10, decaytime=0.01, combfreq=500, amp=1,
        atk=5, rel=5, gate=1, rate=1, durmin=0.05, durmax=0.2, posmin=0, posmax=1;
        var sig, env;
        sig = GrainBuf.ar(
            2,
            Dust.kr(density),
            LFNoise0.kr(500).exprange(durmin,durmax),
            buf,
            rate,
            LFNoise0.kr(500).range(posmin,posmax),
            2,
            0,
        );
        sig = sig + CombL.ar(sig, 1, 1/(combfreq.lag(0.02)),decaytime);
        sig = LeakDC.ar(sig);
        sig = HPF.ar(sig, 10);
        sig = sig * 0.5;
        env = EnvGen.kr(Env.new([0,1,1,0],[atk,0.01,rel],[1,0,-1],2),gate,doneAction:2);
        sig = sig * env * amp;
        Out.ar(out,sig);
    }).add;
});
)

(
//empty event dictionary
~events = Dictionary.new;

//1
~events.add(\cmajorscale1 -> {
    ~cmajorscaleSynth1 = Synth.new(
        \flanger,
        [
            \buf, ~whiddittwoM1.bufnum,
            \density, 50,
            \atk, 0.1,
            \posmin,0,
            \posmax, 1,
            \amp, 0.25,
        ]
    );
});
//2
~events.add(\cmajorscale1_startControl -> {
    Tdef(\cmajorscale1FX, {
        loop{
            ~cmajorscaleSynth1.set(
                //\amp, d[\a].linexp(~a_min,~a_max,0.05,0.45),
                //\decaytime, d[\c].linexp(~b_min,~b_max,10,3),
                \combfreq, ([60,62,64,65,67,69,71,72]).at(d[\b].linlin(~b_min,~b_max,0,7).round)
            );
            0.05.wait;
        }
    }).play
});
//3
~events.add(\cmajorscale1_stopControl -> {Tdef(\cmajorscale1FX).stop});
//4
~events.add(\cmajorscale1_fade -> {~cmajorscale1.set(\gate, 0)});


//1
~events.add(\cminorscale1 -> {
    ~cminorscaleSynth1 = Synth.new(
        \flanger,
        [
            \buf, ~whiddittwoM1.bufnum,
            \density, 50,
            \atk, 0.1,
            \posmin,0,
            \posmax, 1,
            \amp, 0.25,
        ]
    );
});
//2
~events.add(\cminorscale1_startControl -> {
    Tdef(\cminorscale1FX, {
        loop{
            ~cminorscaleSynth1.set(
                //\amp, d[\a].linexp(~a_min,~a_max,0.05,0.45),
                //\decaytime, d[\c].linexp(~b_min,~b_max,10,3),
                \combfreq, ([60,62,63,65,67,68,71,73]).at(d[\b].linlin(~c_min,~c_max,0,11).round
            );
            0.05.wait;
        }
    }).play
});
//3
~events.add(\cminorscale1_stopControl -> {Tdef(\cminorscale1FX).stop});
//4
~events.add(\cminorscale1_fade -> {~cminorscale1.set(\gate, 0)});

        Tdef(\getdata).play;
)


~events[\cmajorscale1].value;
~events[\cmajorscale1_startControl].value;
~events[\cmajorscale1_stopControl].value;
~events[\cmajorscale1_fade].value;

~events[\cminorscale1].value;
~events[\cminorscale1_startControl].value;
~events[\cminorscale1_stopControl].value;
```

```
~events[\cminorscale1_fade].value;


//Print Values
x = {{d[\a].postln;0.07.wait;}.loop}.fork;
x = {{d[\b].postln;0.07.wait;}.loop}.fork;
x = {{d[\c].postln;0.07.wait;}.loop}.fork;
x.stop;
```

PDF document made with CodePrint using Prism